Method of Changing the Page Size of a DB2 Table Space While Keeping the Object Available

DESCRIPTION

BACKGROUND OF THE INVENTION

[Para 1] The present invention relates generally to the field of memory allocation and organization in relational databases. More specifically, the present invention is related to reorganizing object memory and increasing page size allocations.

DISCUSSION OF PRIOR ART

[Para 2] As is commonly defined in database applications, a database consists of sets referred to and displayed as tables. In turn, tables consist of records which are referred to and displayed as rows of a table. Lastly, each row consists of individual data fields known as columns. As data fields, rows, and columns are added and deleted from tables, the probability increases for data that is logically sequential in nature to be stored on pages that are physically non-sequential. This leads to sub-optimal performance, as additional read operations become required to access non-sequentially stored data. As tables are updated with deletions and insertions, performance degradation also occurs through the fragmentation of leaf pages, badly clustered indices (i.e. when the physical index page no longer matches the sequence of keys on a current page), and an index developing a less than optimally efficient number of levels. Fragmentation of leaf pages leads to increasing input/output (I/O) costs as more leaf pages must be read to fetch pages corresponding to a single, designated table. Additionally, when leaf

pages are badly clustered, sequential pre-fetching is inefficient and results in more I/O waits.

[Para 3] Furthermore, when an object is created in a database, the page size allocated for the object is chosen as well. As the object increases with size, either through the addition of columns or rows, the initially allocated page may no longer be adequate to store the entire object. As a result, it is frequently necessary to re-allocate a page of appropriate size for a table. Re-allocation of a page requires unloading and reloading data from each individual data field in the table, making new image copies of the table, and re-granting previous authorizations. While these operations are in progress, the table object is unavailable for substantial periods of time.

[Para 4] Prior art attempts of changing an initially allocated page size are recited in patent literature. U.S. Patent application publication 2003/0115220 teaches the editing of a production data store by shadowing content. In this method, changing a page size changes the production data store by making a shadow the permanent production data store and also updating the databases control blocks to reflect the new page size. However, this method is limited in that only the shadow is modified; no change is made to the permanent production data store.

[Para 5] International patent application publication WO 02/098048 discloses a method for performing an online reorganization of a database. However, the method is limited in that it does not describe changing the attributes of the physical object.

[Para 6] U.S. patent application publication 2003/0130985 discloses a method for upgrading a database with a shadow system. This method references two separate systems operated in parallel; the shadow system is updated via trigger reports from the main system. The method for changing page size only works on a single system. Processing occurs on an original page set in the main system until a switch to the shadow system occurs. Therefore, if updates are allowed during the reorganization process, then both original page set and shadow are updated.

[Para 7] As is shown, prior methods of changing page size fail to teach or suggest a method for dealing with the problems and performance degradation associated with fragmentation and badly clustered indices.

[Para 8] Whatever the precise merits, features, and advantages of the above cited references, none of them achieves or fulfills the purposes of the present invention.

SUMMARY OF THE INVENTION

[Para 9] The method of the present invention provides for the reorganization of a table space to improve access performance and reclaim fragmented space. A database reorganization utility (hereafter, DB2 REORG) is utilized to write rows that are added to a designated object, to a larger page. In this manner, the designated object would remain available, would not require recreation, nor would existing DB2 authorizations be deleted. By reorganizing table space such that constituent rows are read from existing pages and then copied to larger pages, which will subsequently be externalized, the requirement to take the designated object offline while changing the page size is obviated. Subsequent to table reorganization, DB2 control blocks along with the DB2 catalog are updated to reflect the change in page size.

[Para 10] An exemplary embodiment comprises the steps of: (a) blocking write access to data being reorganized; (b) identifying object table spaces that are related to the table space being reorganized; (c) concurrently creating a shadow data set for each of the object table spaces and a shadow data set for the table space and associated indexes; (d) loading rows into shadow data sets, and for each row loaded, reading objects from each of object table spaces relating to a loaded row and writing the read object to a corresponding shadow data set; (e) switching original data set with shadow data sets; and (f) allowing write operations related to data being organized to proceed.

[Para 11] In using the DB2 REORG warehouse utility to increase page size, a decrease in performance degradation due to fragmentation is also affected. Database reorganization utilities allow one to rearrange a table in physical

storage, thereby eliminating fragmentation and ensuring efficient storage in a database. The DB2 REORG utility is also used to compact data and control the order in which constituent rows of a table are stored, in most cases according to an index value on physically contiguous pages. During the reorganization process, information about the current progress of table reorganization is written to the history file for database activity. The history file contains a record for each reorganization event. Additionally, table snapshots are used to monitor the progress of reorganization. In a production environment, data remains available and users maintain read and write access to a designated table while indices are being rebuilt.

[Para 12] The DB2 REORG utility is such that changes made to an underlying table with the potential to affect indices while the reorganization is in progress, are logged. In addition, changes made during reorganization are placed in internal memory buffer space, if any such memory space exists. The internal memory buffer space is a designated memory area allocated on demand from the utility heap to store the changes to the object being created or reorganized. The use of the memory buffer space allows object reorganization by initially reading directly from memory, and if necessary, by later accessing change logs. Allocated memory is freed upon completion of reorganization. Subsequently, the DB2 reorganization utility processes change logs to maintain current writing activity.

[Para 13] Additionally, the method of the present invention occurs in a single step; as additional data (e.g., rows or columns) are written to an object (e.g., a table) it is also being simultaneously written to a larger page. Data comprising the designated object is therefore available for read access throughout a majority of the process.

BRIEF DESCRIPTION OF THE DRAWINGS

[Para 14] Figure 1 illustrates a prior art method of changing the page size of an existing DB2 table space.

[Para 15] Figure 2 illustrates the method of the present invention for changing the page size of an existing DB2 table space.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

[Para 16] While this invention is illustrated and described in a preferred embodiment, the invention may be produced in many different configurations. There is depicted in the drawings, and will herein be described in detail, a preferred embodiment of the invention, with the understanding that the present disclosure is to be considered as an exemplification of the principles of the invention and the associated functional specifications for its construction and is not intended to limit the invention to the embodiment illustrated. Those skilled in the art will envision many other possible variations within the scope of the present invention.

[Para 17] The current DB2 REORG utility works by: allocating "shadow" data sets for each table space and its indexes; blocking write access to the data; unloading rows from the original tables space; sorting the rows; loading the rows into the shadow data sets and extracting index keys for each row as it is loaded; sorting the index keys; building the indexes from the sorted index keys; switching the original data sets with the shadow data sets; and allowing write operations to proceed.

[Para 18] Shown in figure 1 is the prior art method of changing the page size of an existing a DB2 table space is shown. In step 100, individual data fields from a designated object from a currently selected table space is unloaded. Data in these fields are held until they are written to another page. To "drop", or reclaim the memory allocated for the currently selected table space in step 102, it is necessary to identify existing authorization for tables in the currently selected table space and also to identify existing indices on tables in the currently selected table space. Subsequently in step 104, the table space is re-created in a buffer corresponding to a larger page size. Tables that originally existed in the original table space are re-created in the new table space, as are indices and views for each of the tables in the dropped table space in step 106. Authorizations for the newly created tables are granted in

step 108, and in step 110, individual data fields are loaded into the recreated tables. The process completes in step 112 with rebuilding indices for newly reloaded data. In this process, it is important to note that data is unavailable from the time that the table space is dropped in step 102 until after indices are rebuilt in step 112. Additionally, each of these steps must occur sequentially and manually.

[Para 19] Referring now to figure 2, the method of the present invention is shown. The DB2 REORG utility is used to put sorted rows into larger pages and subsequently, to externalize them. The DB2 REORG utility creates shadow data sets for the table space and, at the same time, creates shadow data sets for the indexes. In step 200, the process begins by allocating shadow data sets for each of the designated table spaces and indices. Subsequently, write access to individual data fields of designated table spaces is blocked in step 202. Following, in step 204, shadow control blocks are updated with new page size values. In step 206, rows are unloaded from their original table spaces, sorted, and loaded into shadow data sets; index keys for each row are also extracted as each row is loaded. From index keys sorted in step 206, indices are built in step 208. During the former steps of the method of the present invention, data in table spaces undergoing reorganization is available for reading. In step 210, DB2 control blocks and catalog are updated to reflect the updated page size. Finally, shadow data sets are switched original data sets in step 212, and write operations are allowed to proceed.

[Para 20] Additionally, the present invention provides for an article of manufacture comprising computer readable program code contained within implementing one or more modules to update page size concurrent with the reorganization of database table space. Furthermore, the present invention includes a computer program code-based product, which is a storage medium having program code stored therein which can be used to instruct a computer to perform any of the methods associated with the present invention. The computer storage medium includes any of, but is not limited to, the following: CD-ROM, DVD, magnetic tape, optical disc, hard drive, floppy disk, ferroelectric memory, flash memory, ferromagnetic memory, optical storage,

charge coupled devices, magnetic or optical cards, smart cards, EEPROM, EPROM, RAM, ROM, DRAM, SRAM, SDRAM, or any other appropriate static or dynamic memory or data storage devices.

[Para 21] Implemented in computer program code based products are software modules for:

[Para 22] (a) allocating shadow data sets for each of the table spaces and indexes; (b) blocking write access to the data; (c) updating allocated shadow data sets with new page size values; (d) unload rows from the original table spaces; (e) sorting and loading the rows into the shadow data sets, extracting index keys for each row as it is loaded; (f) building indices from sorted index keys; (g) updating DB2 control blocks and catalog with new page size values; and (h) switching the original data sets with the shadow data sets.

CONCLUSION

[Para 23] A method has been shown in the above embodiments for the effective implementation of changing the page size of a DB2 table space while allowing the availability of the object. While various preferred embodiments have been shown and described, it will be understood that there is no intent to limit the invention by such disclosure, but rather, it is intended to cover all modifications falling within the spirit and scope of the invention, as defined in the appended claims. For example, the present invention should not be limited by software/program, computing environment, or specific computing hardware.

[Para 24] The above enhancements are implemented in various computing environments. All programming and data related thereto are stored in computer memory, static or dynamic, and may be retrieved by the user in any of: conventional computer storage, display (i.e., CRT) and/or hardcopy (i.e., printed) formats. The programming of the present invention may be implemented by one of skill in the art of object-oriented programming.